

2       What is claimed is:

3  
4       1. A method of synchronizing threads in a multiple thread  
5       system, comprising:

6                 defining an entity which maintains a count of values which  
7       increases the value maintained by the object; and  
8                 defining a check operation for said element in which,  
9       during the checking operation, a calling thread is suspended,  
10      and the check is suspended until the value maintained by the  
11      entity has reached or exceeded a given value.

1       2. A method as in claim 1 which said entity is allowed  
2       only to increment between allowable values, and not to decrement  
3       its value.

1  
2       3. A method as in claim 1 wherein said entity is a  
3       counter that is only allowed to include integers.

1       4. A method as in claim 3 wherein an initial value of the  
2       counter is zero.

1  
2       3       5. A method as in claim 1, wherein said entity is a/are  
3       flags.

1       6. An apparatus comprising a machine-readable storage  
2 medium having executable instructions for managing threads in a  
3 multithreaded system, the instructions enabling the machine to:  
4       define an entity which maintains a count of values and  
5 which is allowed to increment between allowable values;  
6       determine a request for value of the element from a calling  
7 thread; and  
8       establish a check operation for said element in which said  
9 calling thread is suspended until the entity reaches a  
10 predetermined value.

1       7. An apparatus as in claim 6, wherein said entity is a  
2 monotonically increasing counter.

1       8. An apparatus as in claim 6, wherein said entity is a  
2 flag.

1       9. A apparatus as in claim 6 wherein said system has a  
2 plurality of processors therein, wherein each of said processors  
3 is running at least one different ones of said threads.

1       10. A method as in claim 1, further comprising defining an  
2 error for an operation that decreases the value maintained by

3 the object to occur concurrently with any check operation on the  
4 object.

1 11. A method as in claim 1, wherein the value maintained  
2 by the object is a numeric value and the increment operation  
3 increases the value by a numeric amount.

1 12. A method as in claim 1, wherein the value maintained  
2 by the object is a Boolean value or a binary value and the  
3 increment operation is a "set" operation that changes the value  
4 from one state to the other state.

1 13. A method as in claim 2, wherein the value maintained  
2 by the object is a Boolean value or a binary value and the  
3 increment operation is a "set" operation that changes the value  
4 from one state to the other state.

1 14. A method as in claim 12, further comprising  
2 establishing an error for an increment operation on the object  
3 to occur more than once.

1 15. A method of defining program code, comprising:  
2 determining different parts of a program which can be  
3 executed either sequentially, or in multithreaded parallel by

4 different threads, and which has equivalent results when  
5 executed in said sequential or multithreaded parallel; and  
6 defining said different parts as being multithreadable.

1       16. A method as in claim 15 wherein said determining is  
2 based on a set of conditions that are sufficient to ensure the  
3 equivalence of sequential and multithreaded execution of a  
4 program construct.

1       17. A method as in claim 15 wherein said different parts  
2 are defined as being multithreadable using an equivalence  
3 annotation within the program code.

1       18. A method as in claim 17 wherein said annotation is a  
2 pragma.

1       19. A method as in claim 17 wherein said annotation is a  
2 code comment.

1       20. A method as in claim 15 further comprising, within  
2 said code, multithreaded constructs, in addition to said  
3 multithreadable parts.

1        21. A method as in claim 15 wherein said multithreadable  
2 parts includes information which, if executed as threads, will  
3 include the same result as if executed sequentially.

1        22. A method as in claim 15 wherein said part is a  
2 multithreadable block of information.

1        23. A method as in claim 22 wherein said part is a  
2 multihreadable FOR loop.

1        24. A method as in claim 15 further comprising  
2 synchronizing threads using a monotonically-increasing counter.

1        25. A method as in claim 15 further comprising  
2 synchronizing threads using a flag.

1        26. A method as in claim 16, wherein the equivalence  
2 annotation includes a new or existing keyword or reserved word  
3 in the program.

1        27. A method as in claim 16, wherein the equivalence  
2 annotation takes the form of a character formatting in the  
3 program, which can be such as boldface, italics, underlining, or  
4 other formatting.

1           28. A method as in claim 16, wherein the equivalence  
2       annotation takes the form of a special character sequence in the  
3       program.

1           29. A method as in claim 16, wherein the equivalence  
2       annotation is contained in a file or other entity separate from  
3       the program.

1           30. A method as in claim 16, wherein the sequential  
2       interpretation of the execution of the block construct is that  
3       statements are executed one at a time in their textual order,  
4       and the multithreaded interpretation of the execution of the  
5       block construct is that statements of are partitioned among a  
6       set of threads and executed concurrently by those threads.

1           31. A method as in claim 16 further comprising using  
2       monotonic thread synchronization to synchronize actions among  
3       threads.

1           32. A method as in claim 15 wherein:  
2       explicitly multithreaded program constructs are always  
3       executed according to a multithreaded interpretation

1           multithreadable program constructs are either executed  
2       according a multithreaded interpretation or executed according  
3       to a sequential interpretation; and  
4           sequential or multithreaded execution of multithreadable  
5       program constructs is at user selection.

1       33. A method as in claim 32, wherein the sequential or  
2       multithreaded execution of multithreadable program constructs is  
3       signalled by a pragma in the program.

1       34. A method as in claim 32, wherein the method for  
2       selecting sequential or multithreaded execution of  
3       multithreadable code constructs is a variable that is dependent  
4       of the value of a variable defined in the program or in the  
5       environment of the program.

1       35. A method of claim 32 wherein said multiple threaded  
2       construct is a block or for loop.

1       36. A method of coding a program, comprising:  
2           defining a first portion of code which must always be  
3           executed according to multithreaded semantics, as a  
4           multithreaded portion of code;

5 defining a second portion of code, within the same program  
6 as said first portion of code, which may be selectively executed  
7 according to either sequential or multithreaded techniques, as a  
8 multithreadable code construct; and  
9 allowing a program development system to develop said  
10 multithreadable code construct as either a sequential or  
11 multithreaded construct.

1 37. A method as in claim 36, wherein said program  
2 development system includes a compiler.

1 38. A method as in claim 36 wherein said multithreaded  
2 construct defines an operation which has no sequential  
3 equivalent.

1 39. A method as in claim 38 wherein said multithreaded  
2 construct is control of multiple windows in a graphical system.

1 40. A method as in claim 38 wherein said multithreaded  
2 construct is control of different operations of a computer.

1 41. A method as in claim 37 wherein said operation is  
2 executed on a multiple processor system, and different parts of  
3 said operation are executed on different ones of the processors.

1       42. A method as in claim 37 wherein said multithreadable  
2 constructs include a synchronization mechanism.

1       43. A method as in claim 42 wherein said synchronization  
2 mechanism is a monotonically increasing counter.

1       44. A method as in claim 43 wherein said synchronization  
2 mechanism is a special flag.

1       45. A method of integrating a structured multithreading  
2 program development system with a standard program development  
3 system, comprising:

4           detecting program elements which include a specified  
5 annotation;

6           calling a special program development system element which  
7 includes a processor that modifies based on the annotation to

8 form a preprocessed file; and

9           calling the standard program development system to compile  
10 the preprocessed file.

1       46. A method of operating a program language, comprising:

2           defining equivalence annotations within the programming  
3 language which indicate to a program development system of the

4 programming language information about sequential execution of  
5 said statement; and

6 developing the programs as a sequential execution or as a  
7 substantially simultaneous execution based on contents of the  
8 equivalence annotations.

1 47. A method as in claim 46 wherein the equivalence  
2 annotation indicates that the statements are multithreadable.

1 48. A method as in claim 46 wherein the equivalence  
2 annotation indicates that the statements are either  
3 multithreaded or multithreadable.

1 49. A method as in claim 48 wherein said multithreaded  
2 statements must be executed in a multithreaded manner.

1 50. A method as in claim 48 wherein said multithreadable  
2 annotations indicate that the statements can be executed in  
3 either multithreaded or sequential manner.

1 51. A method as in claim 46 wherein said equivalence  
2 annotation is a pragma.

1       52. A method as in claim 46 wherein said equivalence  
2 annotation is a specially-defined comment line.

1       53. A method as in claim 47 further comprising  
2 synchronizing access of threads to shared memory using a  
3 specially defined synchronization element.

1       54. A method as in claim 53 wherein said synchronization  
2 element is a synchronization counter.

1       55. A method as in claim 54 wherein said synchronization  
2 counter is monotonically increasing, cannot be decreased, and  
3 prevents thread operation during its check operation.

1       56. A method as in claim 53 wherein said synchronization  
2 element is a synchronization flag.

1       57. A method as in claim 56 wherein said synchronization  
2 counter is monotonically increasing, cannot be decreased, and  
3 prevents thread operation during its check operation.

1       58. A method as in claim 54 wherein said s counter  
2 includes a check operation, wherein said check operation  
3 suspends a calling thread.

1       59. A method as in claim 58 further comprising maintaining  
2 a list of suspended threads.

1       60. A method of modifying an existing program development  
2 system and environment, comprising:

3       · detecting which components of a program contain  
4 multithreadable program constructs or explicitly multithreaded  
5 program constructs;  
6       · transforming the components of the program that contain  
7 multithreadable program constructs or explicitly multithreaded  
8 program constructs into equivalent multithreaded components in a  
9 form that can be directly translated or executed by the existing  
10 program development system; and  
11              invoking the existing program development system to  
12 translate or execute the transformed components of the program.

1       61. A method as in claim 60, wherein said indicating  
2 comprises giving distinctive names to said component.

1       62. A method as in claim 59, wherein the transforming of  
2 the components of the program that contain multithreadable  
3 program constructs or explicitly multithreaded program  
4 constructs is by source-to-source program preprocessing.

1       63. A method as in claim 61, wherein the result of the  
2 source-to-source program preprocessing is a program component  
3 that incorporates thread library calls representing to the  
4 transformed multithreadable program constructs or explicitly  
5 multithreaded program constructs.

1       64. A method as in claim 63, wherein the thread library is  
2 a thread library designed in part or whole for the purpose of  
3 representing the transformed multithreadable program constructs  
4 or explicitly multithreaded program constructs.

1       65. A method as in claim 63, wherein the thread library is  
2 an existing thread library or a thread library designed for  
3 another purpose.

1       66. A method as in claim 61, wherein the result of the  
2 source-to-source program preprocessing is a program component  
3 that incorporates standard multithreaded program constructs  
4 supported by the existing programming system.

1       67. A method as in claim 59, further comprising renaming  
2 the standard compiler-linker and the standard compiler-linker

3 name is used for a program component transformation tool that  
4 subsequently invokes the renamed standard compiler-linker.

1       68. A method as in claim 59, wherein the operating system  
2 is Linux or another variant of the Unix operating system and the  
3 existing program development environment is the GNU C or C++  
4 compiler or any other C or C++ compiler that operates under the  
5 given variant of the Unix operating system.

1       69. A method as in claim 59, wherein the existing  
2 programming language is a variant of the Java programming  
3 language and the thread library is the standard Java thread  
4 library.

1       70. A method of operating a program operation, comprising:  
2           defining a block of code which can be executed either  
3 sequentially or substantially simultaneously via separate loci  
4 of execution;  
5           running the program during a first mode in said sequential  
6 mode, and running the program during a second mode in said  
7 substantially simultaneous mode.

1       71. A method as in claim 70 wherein said definition is an  
2 equivalence annotation.

1           72. A method as in claim 71 wherein said equivalence  
2 annotation is a pragma.

1           73. A method as in claim 70 wherein, during said  
2 sequential execution, variables are shared.

1           74. A method as in claim 73 wherein said shared variables  
2 can be checked, and operation of check does not suspend  
3 operations of the program.

1           75. A method as in claim 70 wherein during said  
2 substantially simultaneous operations, variables are shared.

1           76. A method as in claim 70 further comprising debugging a  
2 program in said sequential mode and running a debugged program  
3 in said substantially simultaneous mode.

1           77. An object for synchronizing among multiple threads,  
2 comprising:  
3           a special object constrained to have (1) an integer  
4 attribute value, (2) an increment function, but no decrement  
5 function, and (3) check function that suspends a calling thread.

1        78. A method as in claim 77 wherein said check function  
2 suspends a calling thread for a specified time.

1        79. An object as in claim 78 wherein said object includes  
2 a list of thread suspension queues.

1        80. An object as in claim 77 further comprising a reset  
2 function.

1        81. An object as in claim 77 wherein said object is a  
2 counter.

1        82. An object as in claim 77 wherein said object is a flag  
2 having only first and second values.

1        83. A method of integrating a thread management system  
2 with an existing program development system, comprising:

3        first, running a pre-program development system that looks  
4 for special annotations which indicate multithreaded and  
5 multithreadable block of code;  
6        using said special layer as an initial linker; and  
7        then, passing the already linked program to the standard  
8 program development system.

1        84. A method as in claim 83 wherein said program is a C  
2        programming language.